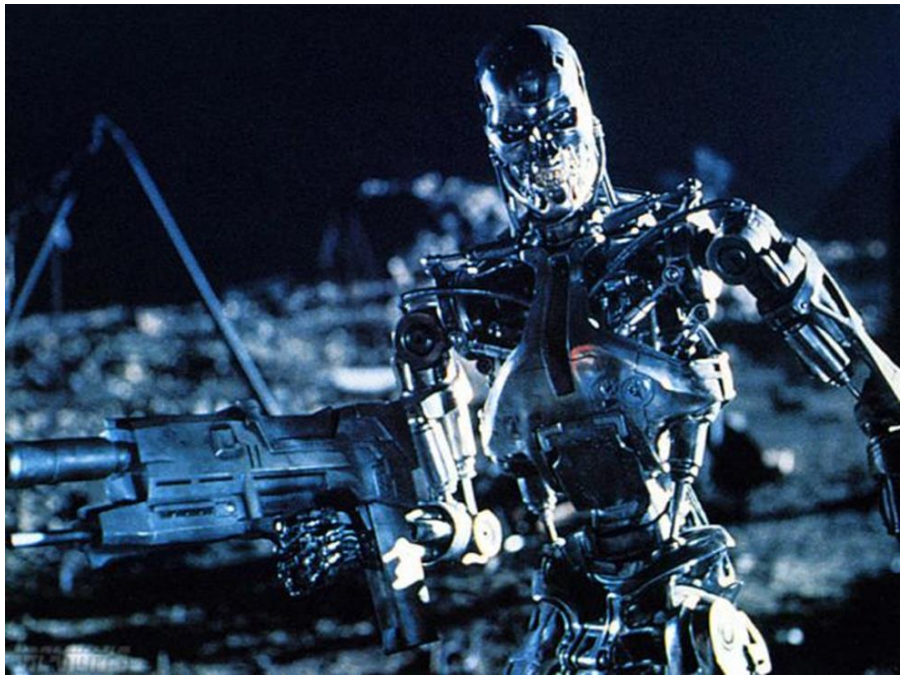
Practical Machine Learning in an Enterprise Setting

Seth Juarez

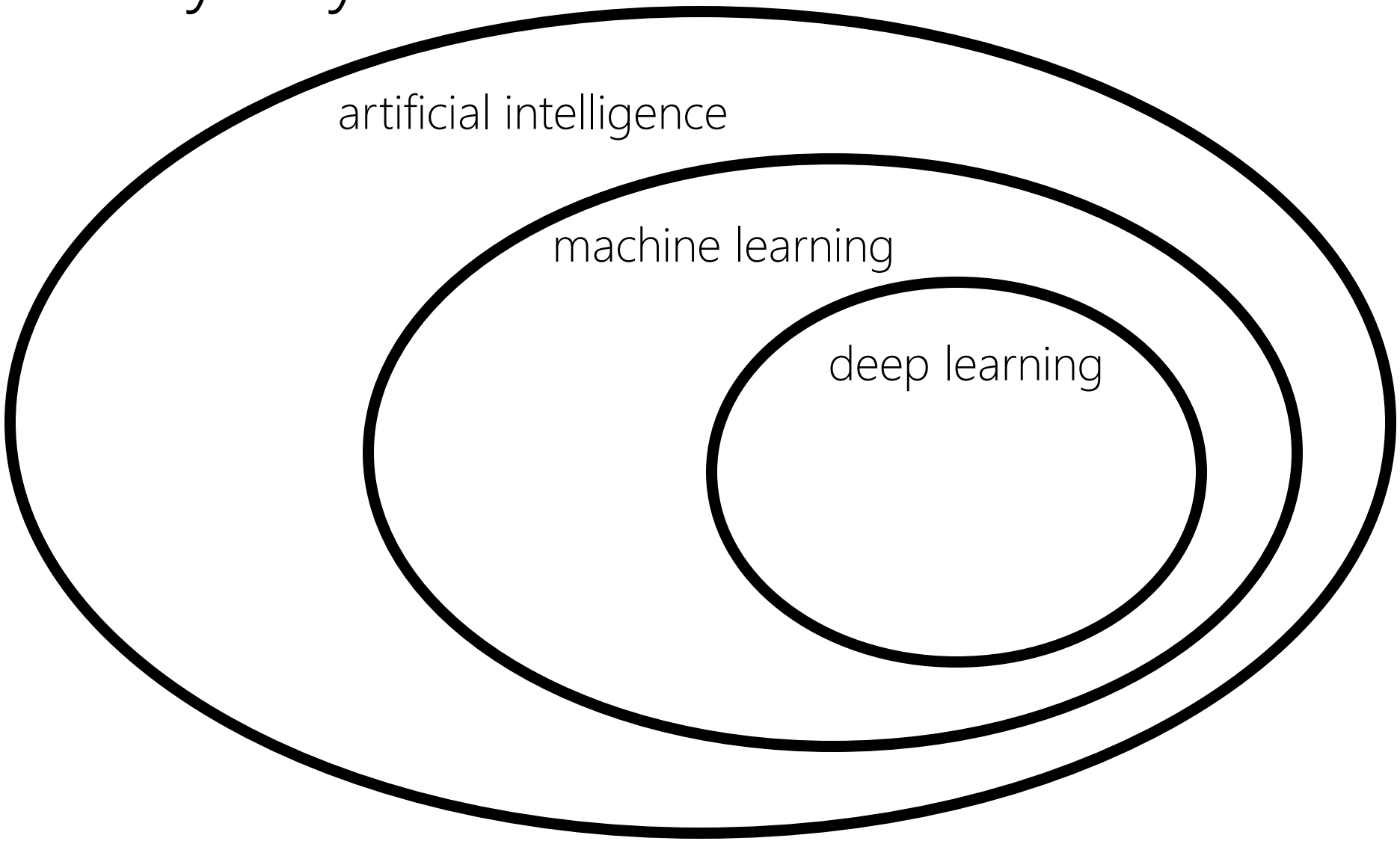
@sethjuarez

Agenda

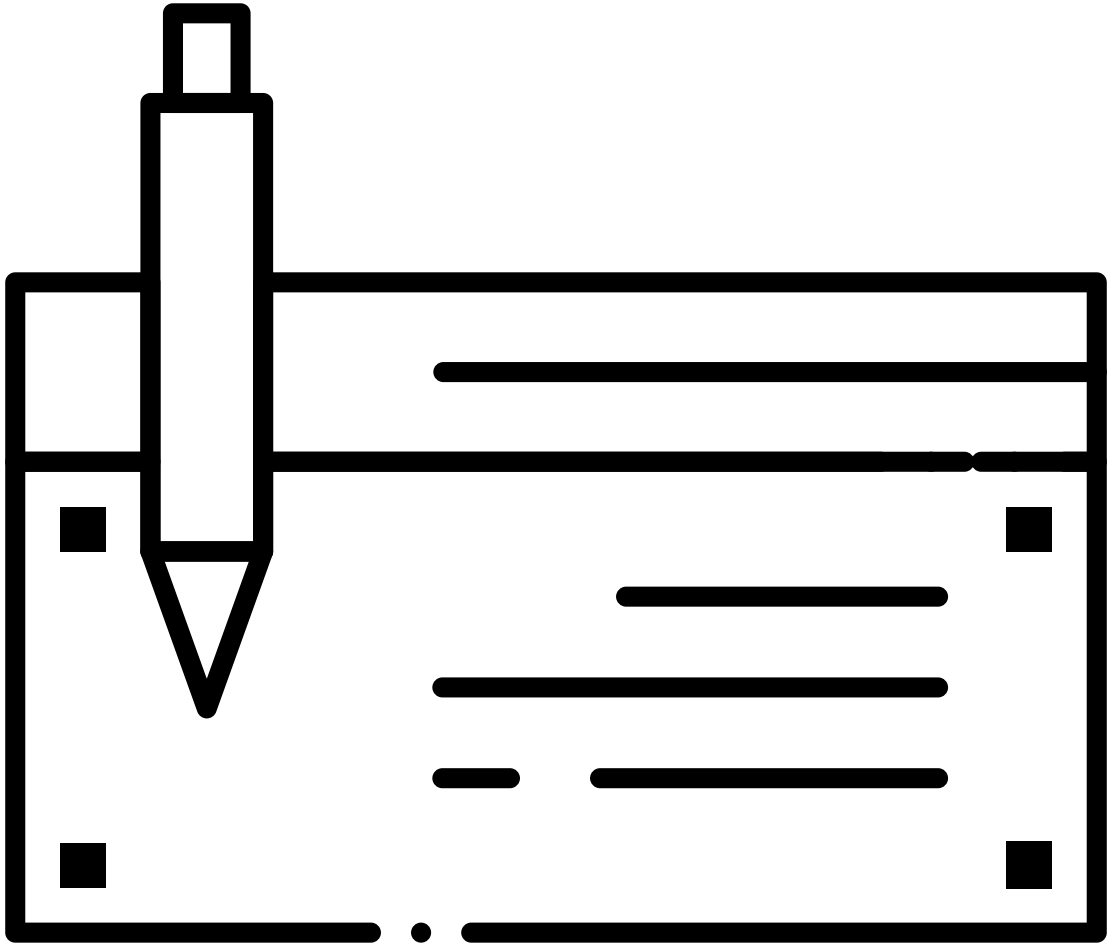
- Introduction to Machine Learning
- Three-pronged strategy
- Cloud based machine learning
- AI as a Team Sport – managing the AI Dev Lifecycle



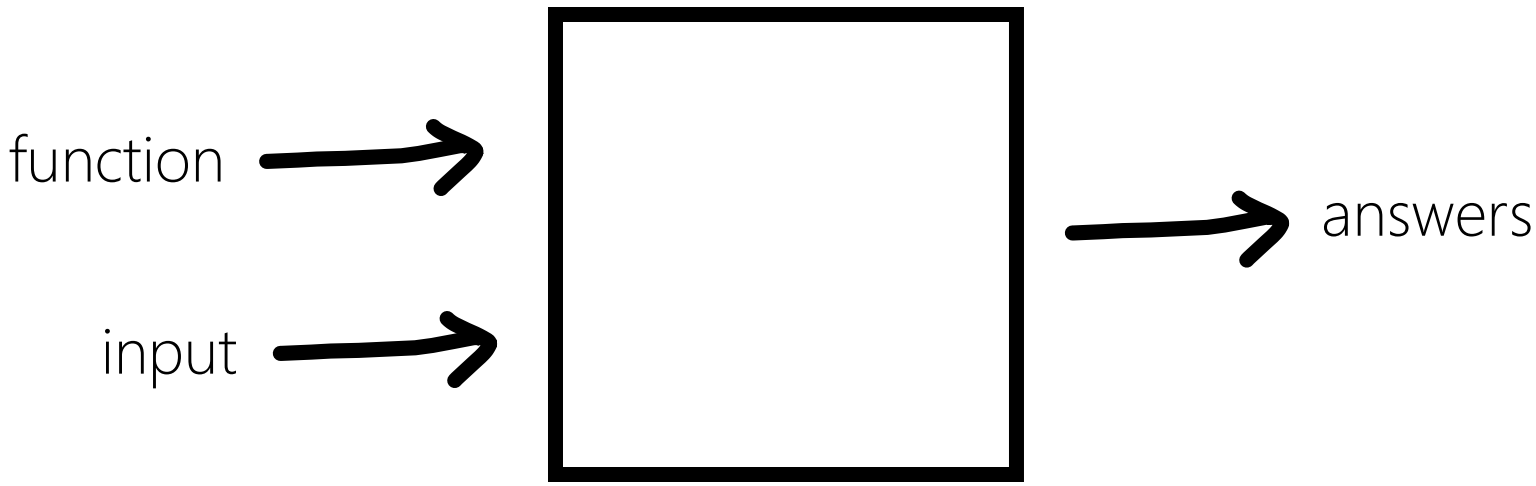
what is it anyway?



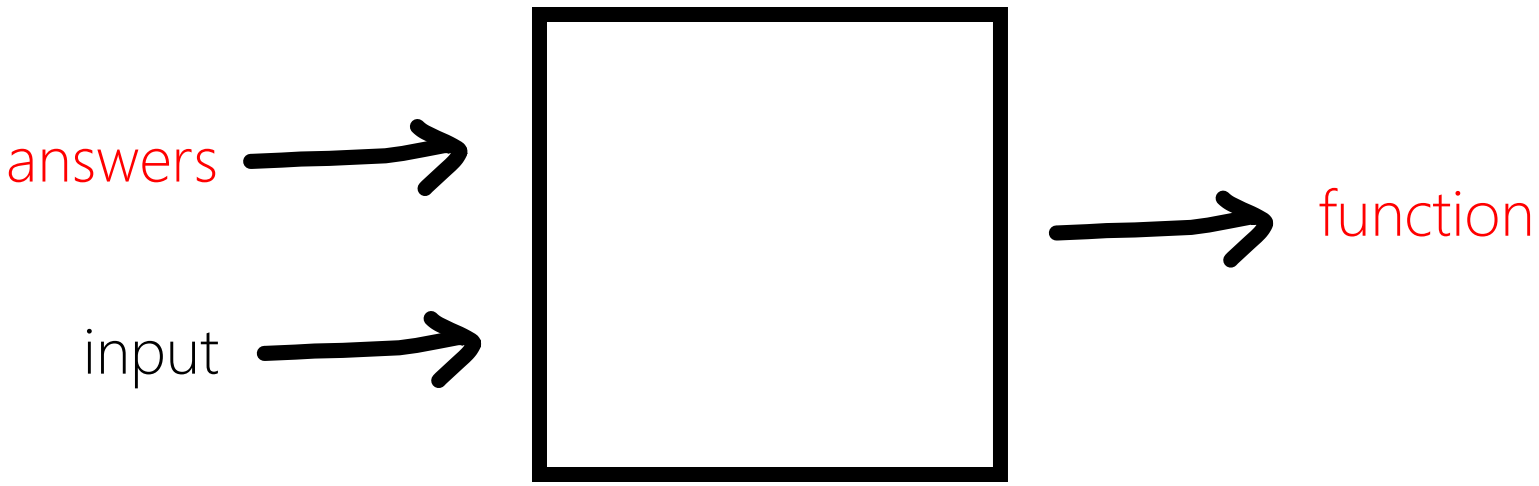
when should I use it?



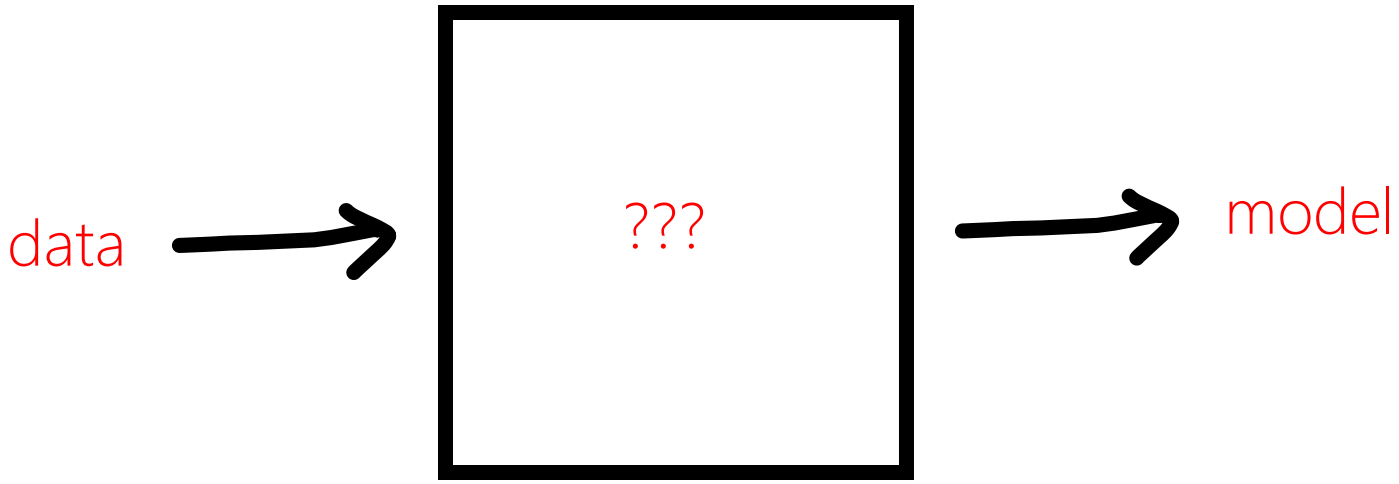
programming



machine learning



machine learning

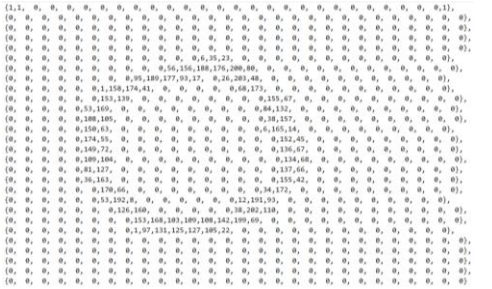


$h(x)$

scikit-learn

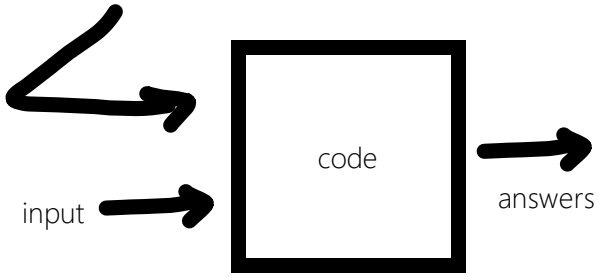
scikit-learn – concepts

- You “pick” the function “shape” – in scikit-learn these are called *classifiers*



- You “fit” the data to the classifier

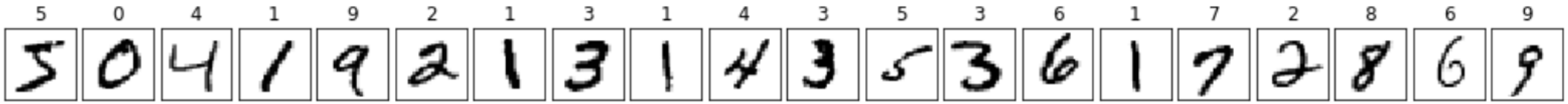
- You use the classifier to predict



scikit-learn – concepts

```
mnist_file = 'https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz'
file, output = urlretrieve(mnist_file, 'mnist.npz')
```

```
with np.load(file) as f:
    x_train, y_train = f['x_train'], f['y_train']
    x_test, y_test = f['x_test'], f['y_test']
```



- You “pick” the function “shape” – in scikit-learn these are called *classifiers*

```
from sklearn import tree
clf = tree.DecisionTreeClassifier()
```

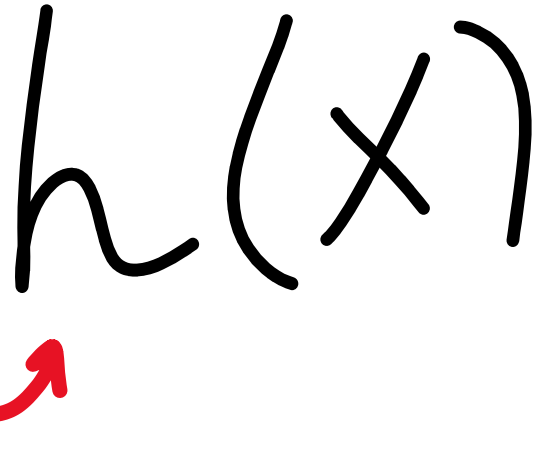
- You “fit” the data to the classifier

```
clf = clf.fit(x_train, y_train)
```

- You use the classifier to predict

```
predictions = clf.predict(x_test)
```

scikit-learn – classifiers



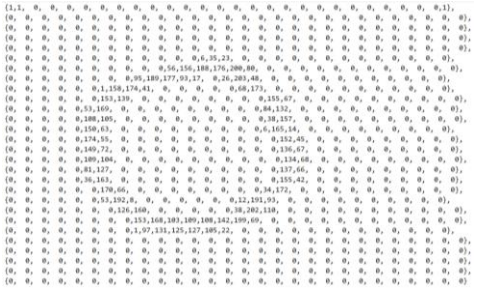
1. Supervised learning

- ▶ 1.1. Generalized Linear Models
- ▶ 1.2. Linear and Quadratic Discriminant Analysis
- 1.3. Kernel ridge regression
- ▶ 1.4. Support Vector Machines
- ▶ 1.5. Stochastic Gradient Descent
- ▶ 1.6. Nearest Neighbors
- ▶ 1.7. Gaussian Processes
- 1.8. Cross decomposition
- ▶ 1.9. Naive Bayes
- ▶ 1.10. Decision Trees
- ▶ 1.11. Ensemble methods
- ▶ 1.12. Multiclass and multilabel algorithms
- ▶ 1.13. Feature selection
- ▶ 1.14. Semi-Supervised
- 1.15. Isotonic regression
- 1.16. Probability calibration
- ▶ 1.17. Neural network models (supervised)

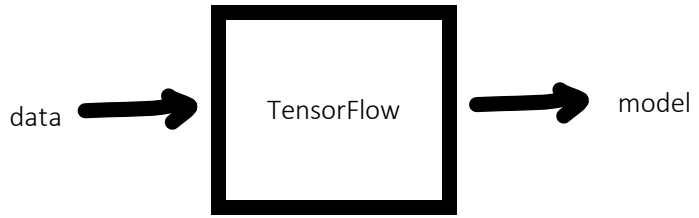
source: https://scikit-learn.org/stable/user_guide.html

TensorFlow – concepts

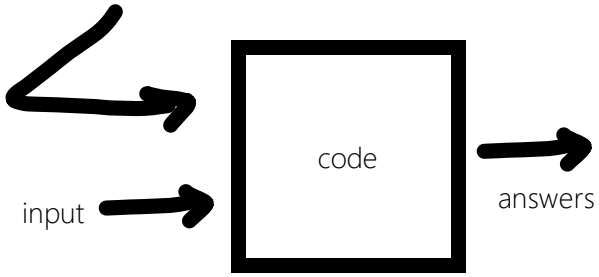
- You “craft” the function “shape” – in TensorFlow this is a *model*



- You “optimize” the model using a *cost/loss function* and *optimizer*



- You use the model to predict



TensorFlow – concepts

- model – the function shape we construct

$$h(x)$$

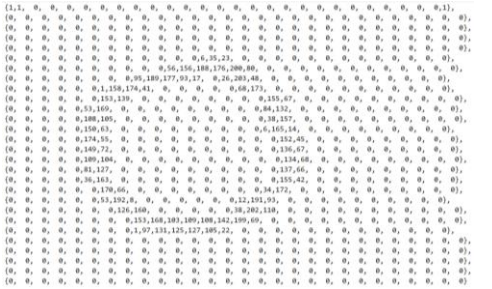
- cost/loss function – a function that tells us how bad we are at predicting

$$l(h(x), y) ?= \emptyset$$

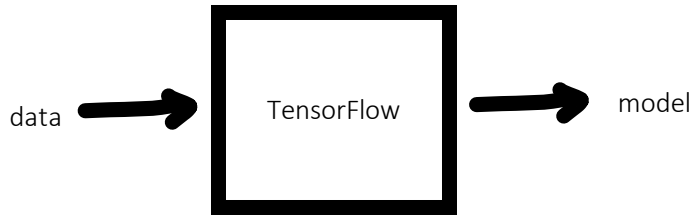
- optimizer – method for reducing how bad we are at predicting

TensorFlow – concepts

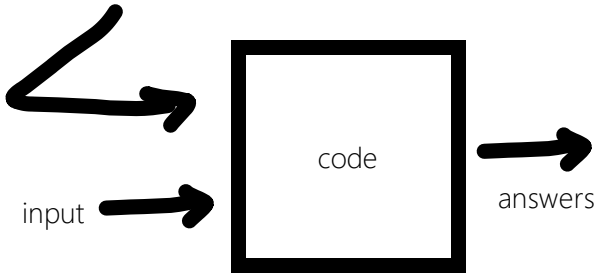
- You “craft” the function “shape” – in TensorFlow this is a *model*



- You “optimize” the model using a *cost/loss function* and *optimizer*



- You use the model to predict



TensorFlow – concepts

- You “craft” the function “shape” – in TensorFlow this is a *model*
- You “optimize” the model using a *cost/loss function* and *optimizer*
- You use the model to predict

```
import tensorflow as tf
from tensorflow import keras
tf.__version__
```

```
# function shape
model = keras.Sequential([
    keras.layers.Dense(128, activation='relu'),
    keras.layers.Dense(10, activation='softmax')
])
```

```
# how to optimize the function
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

```
model.fit(x_train, y_train, epochs=10)
```

```
pred = model.predict(x)
```

TensorFlow – optimizers

- class **Adadelta**: Optimizer that implements the Adadelta algorithm.
- class **Adagrad**: Optimizer that implements the Adagrad algorithm.
- class **Adam**: Optimizer that implements the Adam algorithm.
- class **Adamax**: Optimizer that implements the Adamax algorithm.
- class **Ftrl**: Optimizer that implements the FTRL algorithm.
- class **Nadam**: Optimizer that implements the NAdam algorithm.
- class **Optimizer**: Updated base class for optimizers.
- class **RMSprop**: Optimizer that implements the RMSprop algorithm.
- class **SGD**: Stochastic gradient descent and momentum optimizer.

TensorFlow – loss functions

- class **BinaryCrossentropy**: Computes the cross-entropy loss between true labels and predicted labels.
- class **CategoricalCrossentropy**: Computes the crossentropy loss between the labels and predictions.
- class **CategoricalHinge**: Computes the categorical hinge loss between y_true and y_pred.
- class **CosineSimilarity**: Computes the cosine similarity between y_true and y_pred.
- class **Hinge**: Computes the hinge loss between y_true and y_pred.
- class **Huber**: Computes the Huber loss between y_true and y_pred.
- class **KLDivergence**: Computes Kullback Leibler divergence loss between y_true and y_pred.
- class **LogCosh**: Computes the logarithm of the hyperbolic cosine of the prediction error.
- class **MeanAbsoluteError**: Computes the mean of absolute difference between labels and predictions.
- class **MeanAbsolutePercentageError**: Computes the mean absolute percentage error between y_true and y_pred.
- class **MeanSquaredError**: Computes the mean of squares of errors between labels and predictions.
- class **MeanSquaredLogarithmicError**: Computes the mean squared logarithmic error between y_true and y_pred.
- class **Poisson**: Computes the Poisson loss between y_true and y_pred.
- class **SparseCategoricalCrossentropy**: Computes the crossentropy loss between the labels and predictions.
- class **SquaredHinge**: Computes the squared hinge loss between y_true and y_pred.

TensorFlow – models

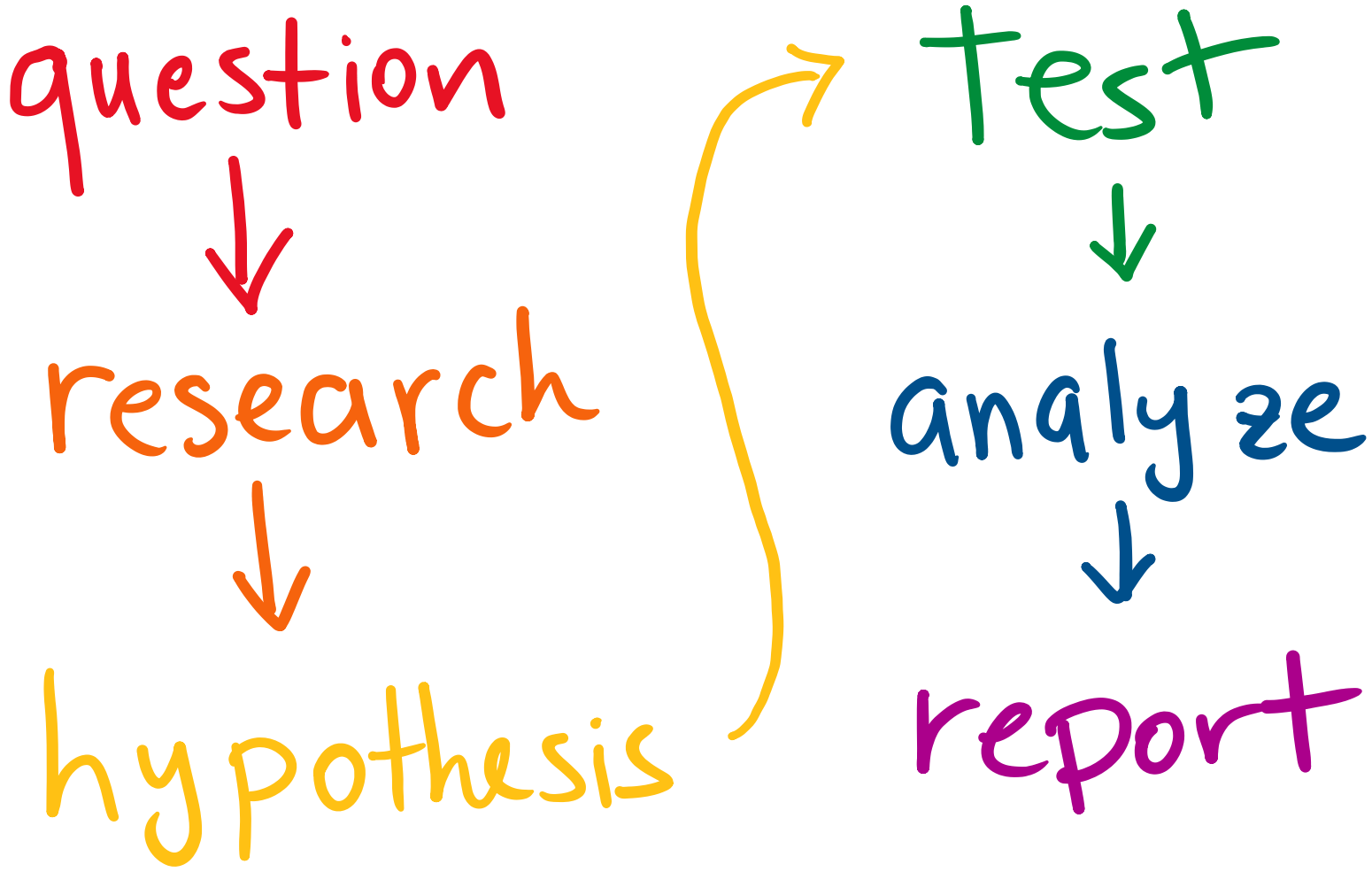
```
# function shape
model = keras.Sequential([
    keras.layers.Dense(layer_width, activation='relu'),
    keras.layers.Dense(10, activation='softmax')
])
```

```
# function shape
model = keras.Sequential([
    keras.layers.Reshape((28, 28, 1)),
    keras.layers.Conv2D(32, (3, 3), activation='relu'),
    keras.layers.Conv2D(64, (3, 3), activation='relu'),
    keras.layers.MaxPooling2D(pool_size=(2, 2)),
    keras.layers.Dropout(0.25),
    keras.layers.Flatten(),
    keras.layers.Dense(layer_width, activation='relu'),
    keras.layers.Dropout(0.5),
    keras.layers.Dense(10, activation='softmax')
])
```

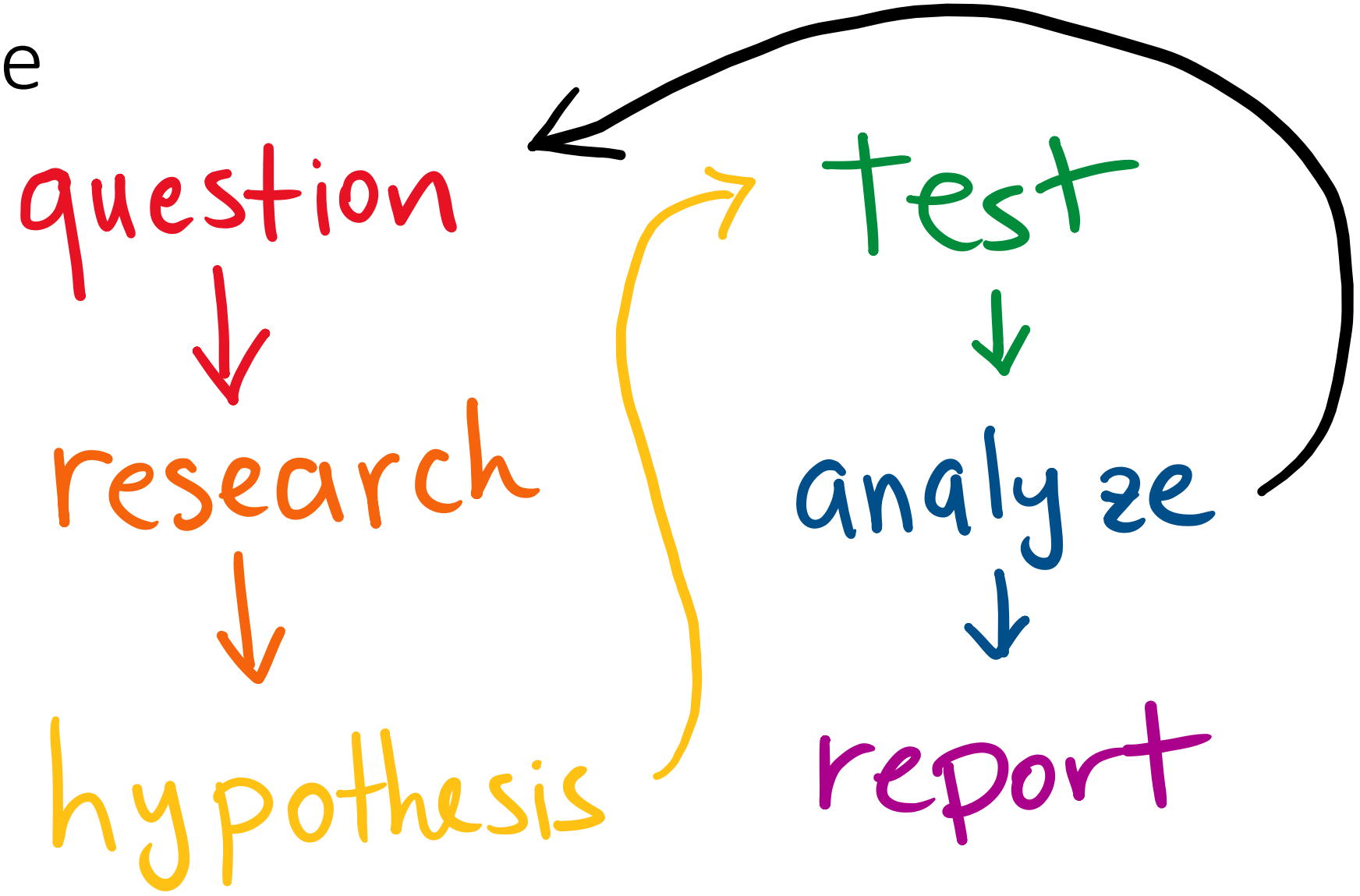
```
base_model = tf.keras.applications.MobileNetV2(input_shape=img_shape,
                                               include_top=False,
                                               weights='imagenet')
base_model.trainable = True

model = tf.keras.Sequential([
    base_model,
    tf.keras.layers.GlobalAveragePooling2D(),
    tf.keras.layers.Dense(1, activation='sigmoid')
])
```

science



science



Three options

- Use pre-built models
- Enhance pre-built models
- Create models

Three options

- Use pre-built models
- Enhance pre-built models
- Create models

No-code/low-code
Cognitive Services

Custom models
AML service

Cognitive Services

… a demo

Custom Vision

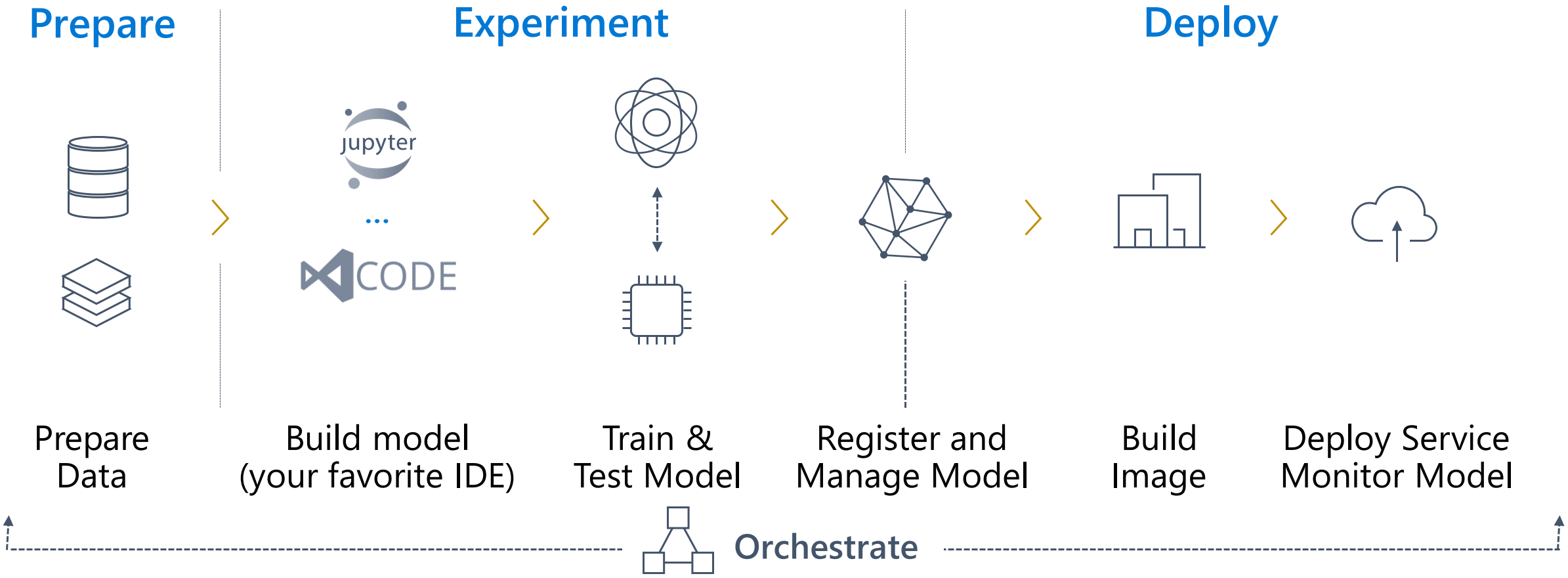
… a demo

Azure Machine Learning service

… a demo

Azure Machine Learning

Typical E2E Process



New
Home
Author
Notebooks
Automated ML
Designer
Assets
Datasets
Experiments
Pipelines
Models
Endpoints
Manage
Compute
Datastores
Data labeling

Welcome to the studio!

Create new

Notebooks
Code with Python SDK and run sample experiments.
Start now

Automated ML
Automatically train and tune a model using a target metric.
Start now

Designer
Drag-and-drop interface from prepping data to deploying models.
Start now

My recent resources

Runs

Run number	Experiment	Updated time	Status
19	mnist_lab	December 2, 2019 1:46 ...	Completed
20	mnist_lab	December 2, 2019 1:18 ...	Completed
1	auto_prediction	December 2, 2019 12:5...	Completed
60	seer	November 18, 2019 1:0...	Completed
18	mnist_lab	November 18, 2019 8:5...	Completed

View all experiments

Compute

Name	Type	Provisioning state	Created on ↓
sauron	Machine Learning Com...	Succeeded (1 node)	November 18, 2019 11:36 AM
gandalf	Machine Learning Com...	Succeeded (1 node)	November 11, 2019 1:11 PM

View all compute

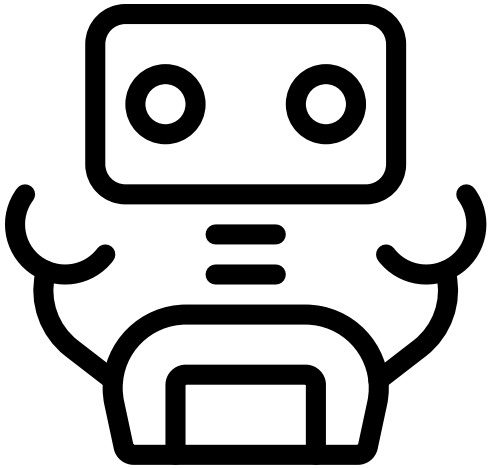
Tutorials

What is Azure Machine Learning?
Train your first ML model with Notebook
Create, explore and deploy automated ML experiments.
What is Azure Machine
What are compute targets in
Deploy models with Azure

Azure Machine Learning

- **Datasets** – registered, known data sets
- **Datastores** – Connections to data
- **Compute** – Managed compute
- **Environments** – defined training and inference environments
- **Experiments** – Training runs
- **Pipelines** – Training workflows
- **Models** – Registered, versioned models
- **Endpoints**:
  - Real-time Endpoints – Deployed model endpoints
  - Pipeline Endpoints – Training workflow endpoints

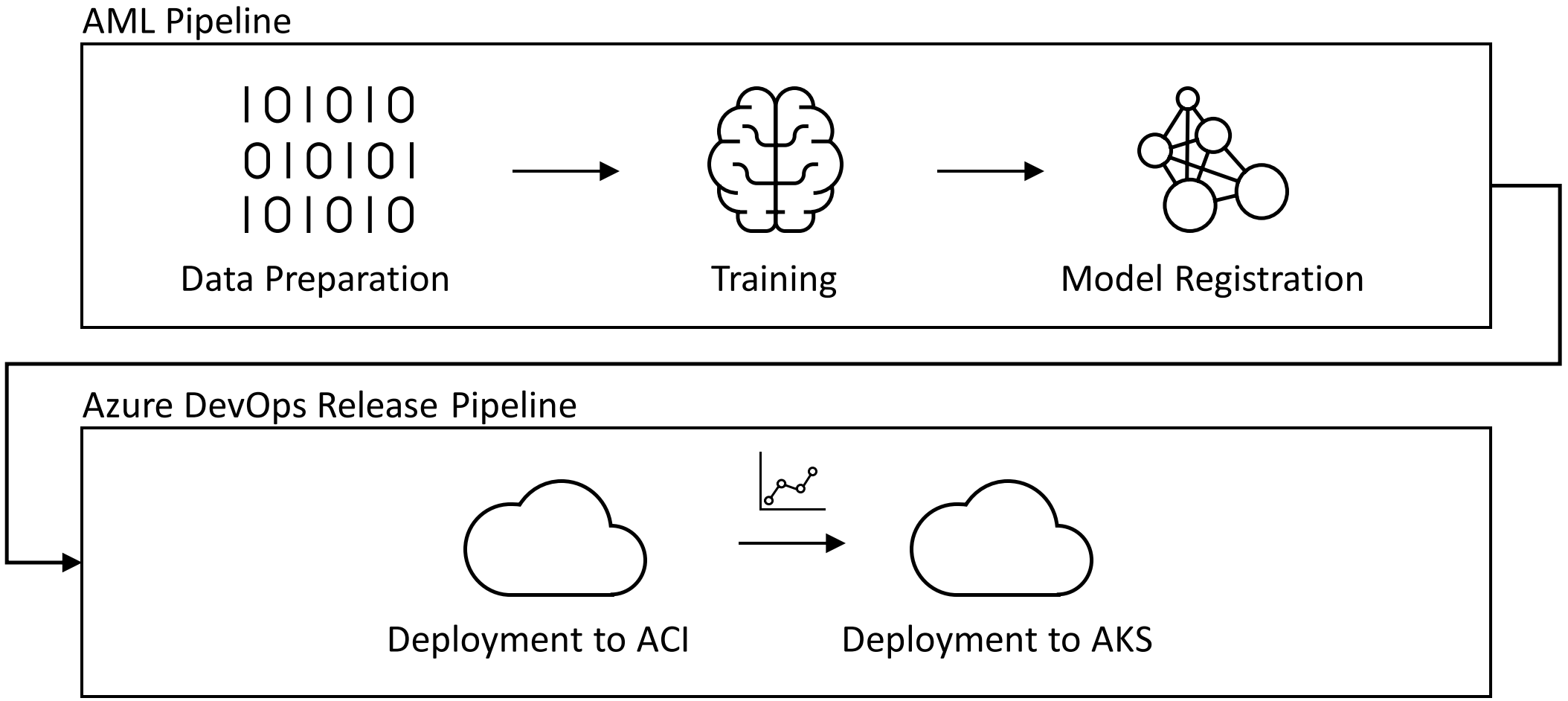
An Important ML Example



burrito

tacos

Complete Pipeline



Ethics

- What is my model doing?
- How do I know it is behaving ethically?

Model Explainability

Review

- Introduction to Machine Learning
  - a function created by data
- Three-pronged strategy
  - prebuilt models
  - customized prebuilt models
  - custom model
- Cloud based machine learning
  - Azure Machine Learning service
- AI as a Team Sport – managing the AI Dev Lifecycle
  - Azure DevOps <-> Azure Machine Learning service
- Ethics
  - model explainability

Questions?

Seth Juarez
Microsoft
@sethjuarez